Hybrid Parallel Programming with MPI and Unified Parallel C

James Dinan, Pavan Balaji, Ewing Lusk, P. Sadayappan, Rajeev Thakur

Overview

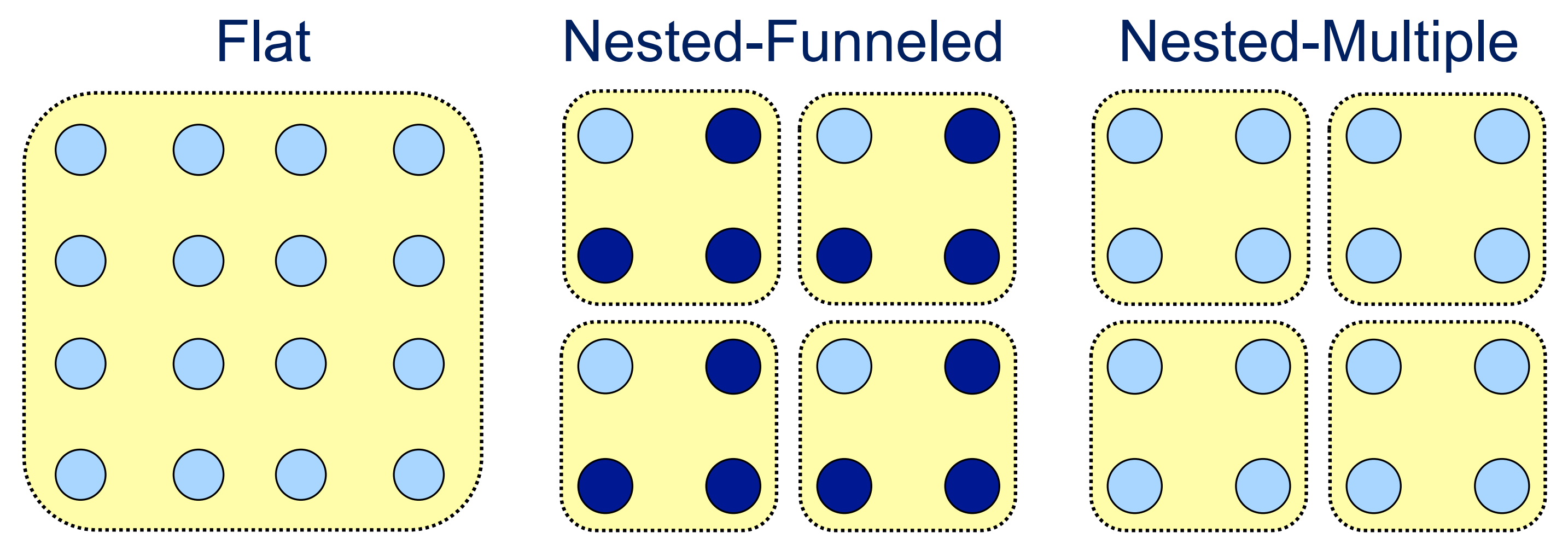
The Message Passing Interface (MPI) is one of the most widely used programming models for parallel computing. However, the amount of memory available to an MPI process is limited by the amount of local memory within a compute node. Partitioned Global Address Space (PGAS) models such as Unified Parallel C (UPC) are growing in popularity because of their ability to provide a shared global address space that spans the memories of multiple compute nodes. However, taking advantage of UPC can require a large recoding effort for existing parallel applications.

In this poster, we describe a new hybrid parallel programming model that combines MPI and UPC. This model allows MPI programmers incremental access to a greater amount of memory, enabling memory-constrained MPI codes to process larger data sets. In addition, the hybrid model offers UPC programmers increased locality control through the creation of static UPC groups that are connected over MPI. As we demonstrate, these groups can significantly improve the scalability of locality-constrained UPC codes.

Advantages of Hybridization

1. **Extend MPI codes with access to more memory:**
   The memory available to an MPI process is limited by the amount of memory in a single node. UPC is able to aggregate the memory of multiple nodes into a single global address space providing access to a greater amount of shared memory.
2. **Improve performance for locality-constrained UPC codes:**
   Some UPC codes lose performance as the number of nodes is increased due to diminishing locality. Hybridization with MPI provides additional locality control and allows the programmer to create static UPC groups that span multiple nodes.
3. **Provide MPI users with an asynchronous global address space:**
   MPI-2 provides one-sided messaging primitives, but they are not quite as flexible as a global address space. MPI targets extreme portability, so it cannot assume that the underlying memory subsystem is coherent. Unfortunately, this results in restrictions on MPI's one-sided semantics includng limitations on local access to shared data, synchronization, and access patterns. UPC harnesses system support to provide a more flexible, convenient, and asynchronous global address space.
4. **Interoperability with libraries like PETSc and SCALAPACK.**

Hybrid Programming Model



○ Hybrid MPI+UPC Process
● UPC Process

Flat: One UPC global address space, all processes can communicate with all other processes using MPI and UPC.

Nested-Funneled: Multiple UPC distributed shared global address spaces connected by MPI. UPC communication is limited to within UPC groups and MPI communication can only be performed by the group masters.

Nested-Multiple: Multiple UPC distributed shared global address spaces connected by MPI. UPC communication is limited to within UPC groups and all processes can communicate via MPI.

Launching Hybrid Applications

Much of the work in hybrid MPI+UPC programming is done by the process manager that launches the application. We've modified Hydra, MPICH2's process manager, to provide additional functionality that allows UPC groups to request more than one MPI rank (`--ranks-per-proc=N`).

Flat: SPMD launch with one MPI task (upcrun) that requests *N* MPI ranks.

```
$ mpiexec --ranks-per-proc=N upcrun -n N myapp
```

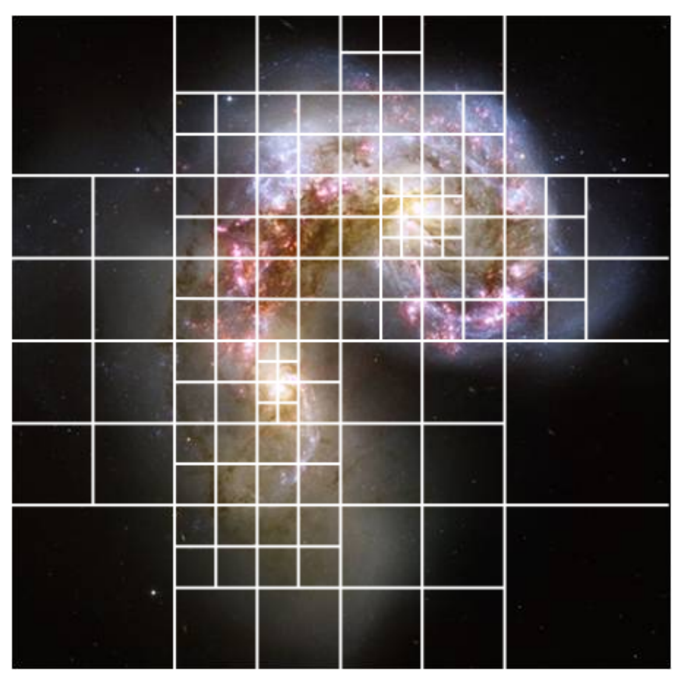
Nested-Funneled: MPMD launch with one task per UPC group

```
$ mpiexec -env HOSTS=hosts.1 upcrun -n N myapp \
  : -env HOSTS=hosts.2 upcrun -n N myapp : ...
```

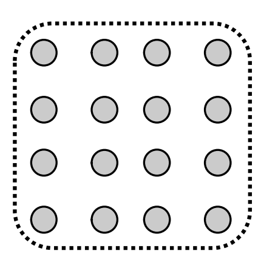
Nested-Multiple: MPMD launch, each UPC group requests *N* MPI ranks.

```
$ mpiexec \
    --ranks-per-proc=N -env HOSTS=hosts.1 upcrun -n N myapp \
  : --ranks-per-proc=N -env HOSTS=hosts.2 upcrun -n N myapp \
  : ...
```

Experimental Evaluation: Barnes-Hut *N*-Body Simulation


Colliding Antennae Galaxies
(Courtesy Hubble Telescope)

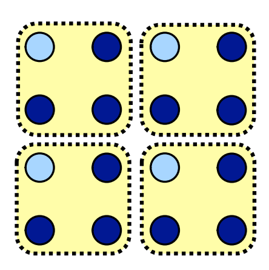
- Simulate motion of *n* bodies over *T* time steps. At each step, calculate the gravitational interaction of each body with all others to find the net force.
- Approximate the interaction between distant bodies as an interaction with the center of mass of whole region.
- Represents sparse volume of 3-dimensional space using a large shared oct-tree (hard to do with MPI, easy to do with UPC).


UPC Only (baseline)

```
for i in 1..t_max
  t <- new octree()
  forall b in bodies
    insert(t, b)
  summarize_subtrees(t)



  forall b in bodies
    compute_forces(b, t)
  forall b in bodies
    advance(b)
```

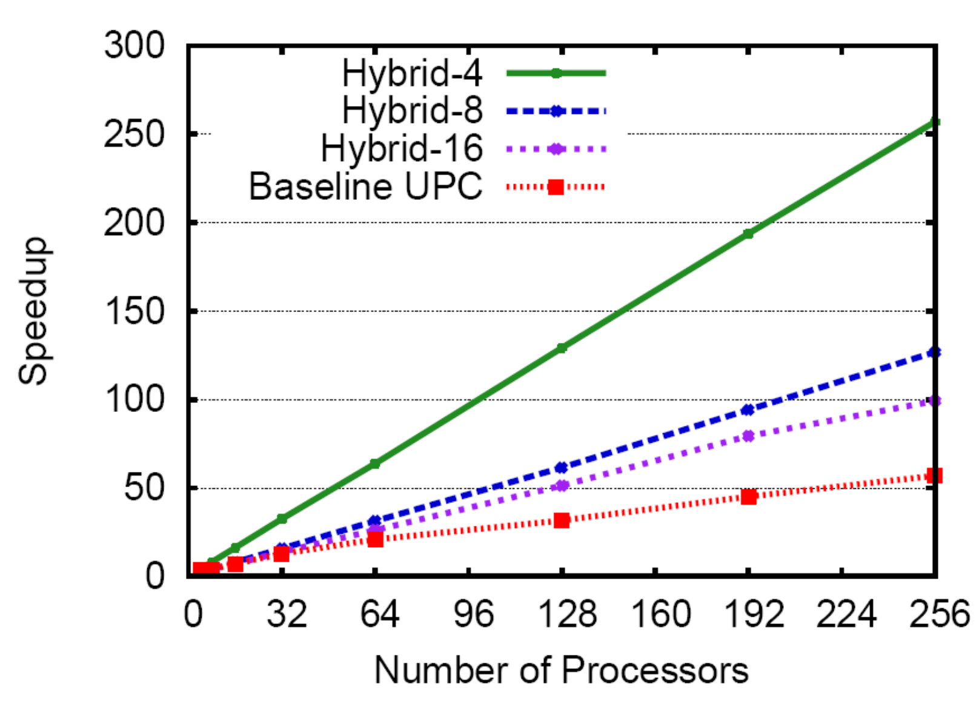

Nested-Funneled Hybrid UPC+MPI

```
for i in 1..t_max
  t <- new octree()
  forall b in bodies
    insert(t, b)
  summarize_subtrees(t)
  our_bodies <-
    partion(group id, bodies)
  forall b in our_bodies
    compute_forces(b, t)
  forall b in our_bodies
    advance(b)
  Allgather(our_bodies, bodies)
```

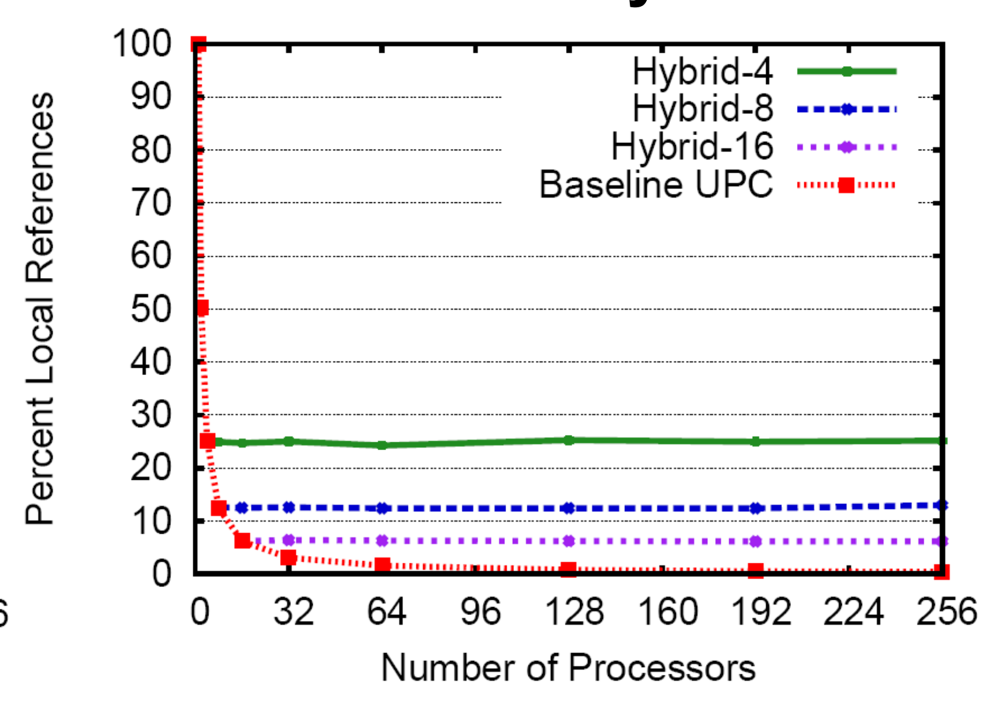
In the nested-funneled implementation, the tree is replicated on every UPC group. New code (highlighted) must be added to distribute work and collect results. In total, of 51 new lines of code (2% increase) were added.

- **Software setup:**
  - GCCUPC compiler; Berkeley UPC runtime 2.8.0, IBV conduit with SSH bootstrap (default is MPI); MVAPICH with Hydra process manager
- **Hardware setup:**
  - Glenn cluster at OSC: 877 Nodes, two dual-core 2.6 GHz AMD Opterons, 8GB RAM per node, and Infiniband interconnect

Performance



Locality



- At 256 processors, baseline UPC speedup is roughly 55x and Hybrid-16 (four quad-core nodes) speedup is 100x.
- The locality plot shows that for the UPC only case, locality diminishes rapidly as the number of processors is increased because the tree is distributed across a larger global address space (more nodes).
- This shows how hybridization provides additional locality control, allowing us to reduce the performance penalty from remote data references.

Conclusions

The hybrid MPI+UPC programming model offers an incremental pathway that allows existing applications to take advantage of MPI's locality control and UPC's global address space. For memory constrained MPI codes, the hybrid model enables computation on larger problems by aggregating the memory of several nodes into a single, shared global address space. For locality-constrained UPC codes, the hybrid model can improve locality through the creation of UPC groups that are connected with MPI.

We have presented an evaluation of this new model on the Barnes-Hut n-body simulation. Compared against a baseline execution on 256 cores, we found that, for groups that span four cluster nodes, the hybrid Barnes-Hut application experiences almost a twofold speedup at the expense of a 2% increase in code size.

For more information, see:

"Hybrid Parallel Programming with MPI and Unified Parallel C." James Dinan, Pavan Balaji, Ewing Lusk, P. Sadayappan, Rajeev Thakur. Proc. 7th ACM Conf. on Computing Frontiers (CF). Bertinoro, Italy. May 17-19, 2010.

Or contact: James Dinan <dinan@mcs.anl.gov>